Eurographics Workshop on Intelligent Cinematography and Editing (2015)
W. Bares, M. Christie, and R. Ronfard (Editors)

Stylistic Patterns for Generating Cinematographic Sequences

Hui-Yin Wu and Marc Christie

INRIA/IRISA Bretagne, Rennes

Abstract

For film editors, the decision of how to compose and sequence camera framings is a question pertaining to a number of elements involving the semantics of shots, framings, story context, consistency of style, and artistic value. AI systems have brought a number of techniques to create procedural generative systems for game animation and narrative content. However, due to its computational complexity, current automated cinematography relies heavily on constraint and rule-based systems, or pre-calculated camera positions and movements that implement well-known idioms from traditional cinematography. Existing dynamic systems only have limited reaction to complex story content and cannot bring affective emotional depth to the scenario. Yet in actual filmmaking, directors often employ camera techniques, which are arrangements of shots and framings, to convey multiple levels of meanings in a sequence.

In this paper we propose a language for defining high-level camera styles called Patterns, which can express the aesthetic properties of framing and shot sequencing, and of camera techniques used by real directors. Patterns can be seen as the semantics of camera transitions from one frame to another. The language takes an editors view of on-screen aesthetic properties: the size, orientation, relative position, and movement of actors and objects across a number of shots. We illustrate this language through a number of examples and demonstrations. Combined with camera placement algorithms, we demonstrate the language's capacity to create complex shot sequences in data-driven generative systems for 3D storytelling applications.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Departing from traditional camera placement and framing problems, virtual cinematography has introduced a camera with no body, no lens, and no complex rails or rigs: an invisible camera that can be duplicated and placed anywhere in the 3D environment and film anything. With the removal of all physical existence of the camera in the scene, what remains of the camera is what appears on the screen: the frames, the shots, the rendering, lights, and colours. This brings the exciting prospect of being able move away from the technical hows of implementing camera positions, and simply discussing the whats in terms of framing and on-screen properties. However this raises a number of questions: do we have sophisticated means to reproduce framings? How much information is "enough" to reproduce a certain framing? And where will we find a vocabulary for style?

Due to the complexity of camera planning problems, existing cinematographic systems often choose rule or constraint-based approaches that define a number of action-camera pairs, that when given an action, a pre-defined camera (whether relative or absolute) is directly placed in the scene [AK05] [BGL98] [BTM00] [LCL*10]. These approaches, with occlusion detection techniques, can provide well-framed camera shots. However, most of them only account for the framing of a single shot and neglect the semantics over multiple shots. Offline approaches can better account for complex story sequences, but are unable to react to dynamic gaming environments [ER07] [GRLC15]. Often the vocabulary of camera idioms defined by these systems are not easily user-extensible. Moreover, the problem of how to place a camera is often approached through a cameraman's point of view by placing a pre-computed camera relative to the environment and targets in order to achieve a certain composition, where in actual practice, the cameraman captures a number of takes, and hands them over to the

film editor to select and sequence. But what if the cameraman could know the exact effect the editor needs in each shot? Take a look at the example shot sequencing in Figure 1 extracted from the movie *The Shining*. The camera crosses the line of interest, making the relative on-screen positions of the actors exchange (crossline), and simultaneously, ends the sequence by cutting the camera consecutively closer over 3 shots (intensify technique). Currently, there is no cinematographic language that (i) is targeted towards this editor's view of formalising complex stylistic properties of framing and sequencing shots, (ii) allows a variety of solutions, and yet (iii) provides sufficient flexibility in the definition of styles.

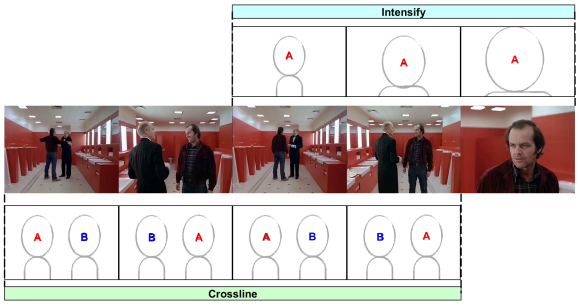

Figure 1: In this sequence from *The Shining*, two techniques–crossline and intensify–are used in an overlapping manner to combine effects from both techniques.

In this paper we propose Patterns, a language for defining complex camera framing and shot sequencing techniques. Patterns represents the semantics of framing and frame sequencing over a number of shots. Patterns can easily define a number of techniques from low-level framing constraints to high-level grammars for shot sequencing. The language describes on-screen properties that the user can easily observe. This approach may depart from traditional descriptions of cinematographic elements, but is a more direct way to discuss the meaning of on-screen movements and the underlying messages they carry.

Thanks to novel camera positioning and transitioning techniques such as [LC12], our language is fully implemented alongside a camera placement technique and shot database, with which we demonstrate a prototype of a data-driven, generative cinematographic system with the patterns as the basis. Using Patterns, we can describe complex constraints on the framing properties on sequences to be generated.

Thus the contributions of this work are (1) a language for describing elements of cinematographic editing style, (2) an implementation of the language that can express complex framing and sequencing properties over a number of shots, and (3) linking of our method to a database of framings and implementing the camera placement in a dynamic 3D virtual environment.

2. Related Work

The identification, analysis, and replication of cinematographic style has much value to both assistive creativity applications as well as cognitive understanding of the effectiveness of filmic style. In this section we observe from film theory, user analysis, and existing computational approaches to the analysis and replication of filmic style.

2.1. Structuring Shot Sequences

When discussing the importance of cinematographic editing and what dynamic storytelling can benefit from classical film editing techniques, [Ron12] reviews recent camera systems in digital game applications. The review is developed from filmic theory, and provides an analysis from Walter Murch's six layers of complexity in camera planning [Mur01] (i.e. 3D space of action, 2D space of screen, eye-trace, rhythm, story, and emotion). The review focuses on techniques for filmic editing in game narratives in order to optimize frame-to-frame transitions and shot technique decisions. This perspective has the strength of maximizing the reaction of the camera to dynamic environments. However, the scope limits the storytelling power and emotional power layers, given that in a computational system, what can be easily modelled may be limited to bottom-up framing, camera movement, or cut type organization. These techniques may be sufficient to convey a scene, yet according to Murch [Mur01], a good cinematographic arrangement should "advance" the story, and conveyance of story and emotion elements accounts for almost three quarters of a good cinematographic arrangement.

Maybe surprisingly, one aspect of the story-discourse axis that is often neglected, and is especially important in interactive storytelling, is that discourse does not only derive from story. Discourse also contributes to story. Analysis of visual narratives such as comic strips suggest that viewers construct logical links between frames as well as structural narrative arcs over the entire sequence [Coh13]. In well-structured sequences, the viewer is found to have the capacity to identify large sections of plot elements such as establishing, initial, prolongation, peak, and release. Moreover, some sections can also be recursively broken down to smaller units, creating complex narratives, or even left out, without interfering with the viewer's comprehension. On the other hand, they can also be misled or confused when elements are switched around or if the narrative is not well-structured.

In the same way, sequencing of shots in a film also pertains to the visual narrative. The design of Patterns is therefore with a mind to create a language that can provide structural semantics to filmic sequences in terms of on-screen framing properties and constraints. This would allow the design of much more complex and meaningful camera styles that can best express story context.

2.2. Effect of Camera Style Aesthetics on Viewers

Previous work has observed how framing techniques such as over-the-shoulder, closeups, long shots, or camera movements provide certain interpretations [Zet07]. What interests us is how sequences of shots with well-designed cinematographic framings can also point to specific interpretations. For example, [CBL11] analyses how the change of distance between shots creates emotional arousals among viewers. Other elements such as tempo [ADV02], content saliency [XWH*11], and aesthetic qualities such as colour or brightness [TSW*14].

In order to provide new methods to annotate and analyze camera styles, Ronfard [RBLA13] designed the prose storyboard language (PSL) as a context-free language for film annotations. The language was in turn used in the work of [GRC*14]. While both Patterns and PSL are targeted towards the description of on-screen properties, their annotation structure, target usage, and implementation vary greatly. First of all, PSL is limited to describing individual shots in a storyboarding fashion and not the relation between shots. On the other hand, Patterns annotates stylistic features of sequences, also emphasizing transitions and relations between framings in a sequence. Thus, while PSL targets a more preproduction phase, Patterns is targeted towards automating the editing phase in virtual cinematographic systems. On the imlpementation, PSL searches for camera positions that fulfills on-screen constraints for each shot separately, whereas Patterns strongly relies on data-driven approaches to find and match both framing and sequencing constraints as an integrated stylistic choice.

Drawing attention to the analytical properties of cinematographic style, it is currently very difficult to conduct evaluations on singular aspects of films style due to the lack of formalisation of style definitions. We thus re-emphasise the importance of ensuring that the pattern language can serve as a formalisation of camera styles.

2.3. Rule and Constraint-Based Systems

An efficient way to view camera sequences is through rule or constraint-based systems, where given a number of parameters on the context, location, action taking place, a shot is selected and the camera is placed. The rule and constraint-based system was first developed by [DZ94] for navigation in 3D environments, and followed by [AK05] [BGL98] [BTM00] with a gradual move towards interactive and storytelling applications. This effectively simplifies the problem of dealing with complex environments. Yet this approach is not adaptable to complex story sequences where multiple styles may overlap in order to achieve an accumulated effect, and the limited extensibility of the vocabulary makes defining new styles difficult. Moreover, sometimes style only refers to certain qualities of the framings, such as size and position of the target on screen, while other characteristics, such as movement or tempo, should be given wider possibilities to explore different framings.

What we also observe from real films is a tendency to reuse classical sequencing techniques across different films. These "patterns" of style can thus be an invaluable asset in dynamic 3D storytelling applications in order to express depth in the story events, which is currently difficult in a rule-based system.

3. Pattern Language

In this section we describe the grammar and semantics of Patterns, including an overview of the basic set of vocabulary, and the logic to construct techniques over editing style using Patterns.

The pattern language is composed of three levels of specifications: framing, operations, and patterns.

3.1. Framing

Framing refers to the most basic unit in cinematography: the frame, with all its components and properties. Limits on framing can be either relative (to previous framing) or absolute. When specifying properties of framing, Patterns often describes the property in relation to some on-screen target, whether an object, actor, or location.

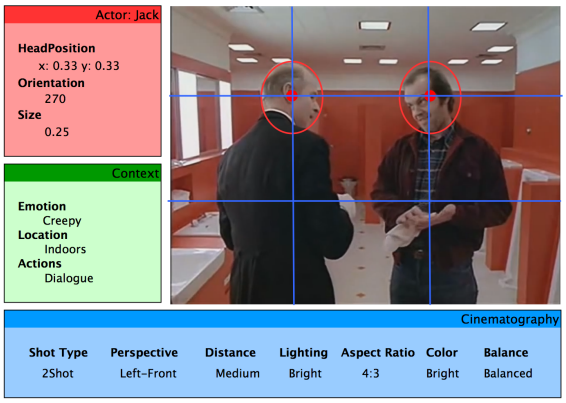

Figure 2: Example properties of annotation on the framing.

Below are the properties for describing framings:

Relative to the same target in the previous framing

1. *position* on the screen, with values of *left*, *right*, *up*, *down*, *forward*, *backward* (in meters)
2. *type* of shot, with optional value of *same*
3. *continuity* between two framings, with values of *match_index_vector* or *match_action*
4. *distance* of the target with values of *closer* or *further*

Absolute values of a target in the current framing

1. *position* on the screen, with absolute values of x and y coordinates
2. *region* on the screen, which can either be a 4- or 9-split screen, with values $R4(1\_1)$ or $R9(1\_1)$ (upper left) to $R4(2\_2)$ and $R9(3\_3)$ (lower right) respectively
3. *type* of shot, with values of *any*, or specific shot types such as *over_the_shoulder*, *2_shot*, or *point_of_view*
4. *angle* of the shot, with values for *horizontal*, *vertical*, and/or *roll* (in radians, 0 being right in front of the target with no roll or tilt)
5. *distance* of the target, with a value on the scale of 1-10 from *extreme_closeup* to *establishing_shot*

Figure 2 shows how a framing can be specified from a real movie scene. Our film annotation tool allows immediate transformation of these on framing properties into a file that can be read as a framing database. The framing properties above currently have dual properties of being (i) annotative: in that it can be used to describe on-screen properties of framings; and (ii) generative: where the properties are used as constraints on framings when defining a camera style. The usage of the framing properties as constraints is illustrated in the next section, when we introduce "operations" as a means of defining stylistic constraints on framing for shot sequences.

We currently use the language to describe framing constraints for two on-screen targets, which can be separate actors and/or objects, or two different targets on the same actor/object (e.g. the two eyes of an actor, ot the hand of one actor and an object in the scene). Though Patterns does not restrict the number of targets, [LC12] demonstrates how it is difficult to guarantee the quality of the shot for framings with more than two characters.

3.2. Operations

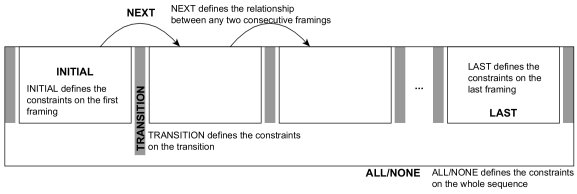

Figure 3: The operationslisted in a pattern sets constraints on certain framings within the sequence.

Stories have an order in which they are presented, and thus the style in which they are presented should also indicate the semantics of sequencing and time. Operations are sequencing constraints on the editing for the framing properties mentioned above. There are 7 types of operations that specify what kind of sequencing constraints the operation reinforces, and an optional *duration* parameter in seconds. The types of operations available in Patterns can be seen in Figure 3, with detailed descriptions below:

All : All selected framings in the pattern have a specified property, or embedding patterns. Example: *All* selected framings have property *distance* ==‘*MLS*’ on *target*1

None : None of the selected framings in the pattern have a specified property. Example: *None* selected framings have $number\_of\_targets < 2$

Initial : the initial selected framing(s). Example: *Initial* selected framing have property *shottype* ==‘*point_of_view*’

Next : with the Initial framing as a basis, specify the relations between any two framings in the pattern. Example: Every *Next* framing after *Initial* have property of *closer* on *target*1

Last : the exiting framing(s) of the pattern.

Ordered : ordered properties (with an integer id for the order) for each framing in the pattern.

Transition style between the framings: can be a shot transition (i.e. *cut*, *fade*, *jump*, *mosaic*), cut condition (*duration*, *target_change*), or movement type (e.g. *pan*, *dolly*, *track*).

We show two example operations in Figure 4: an initial operation which specifies a initial frame with a constraint on the distance property, and transition condition of decreasing the distance property of the framing.

3.3. Patterns

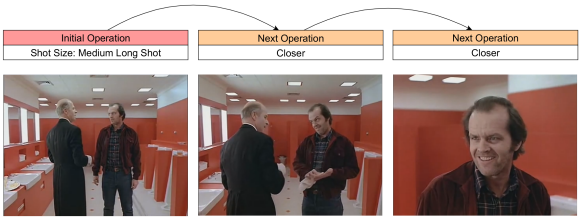

Figure 4: The *initial* operation specifies a frame of size *medium_longshot* (MLS) while the *next* operation of *closer* specifies that the next framing must have a shot distance that is closer than the current framing. The two operations put together would result in the sequence of shots to move closer and closer to the actors starting from a shot of *medium_longshot* (MLS) size. As shown in the screen captures, this makes the target actors in the frame closer and larger.

As shown in Figure 4, when operations are assembled, they can define a way of building and editing shots. This assembling encodes a film style, and is what we call "patterns". Each pattern also takes a number of parameters, including:

Name a title for the pattern

Targets list of targets that operations use as reference to realise framings

Length over a specified number of shots, which can be greater, equal, or less than a certain value

A pattern can also embed other patterns so as to form complex techniques involving a number of consecutive or overlapping layers of style.

In the example shown in Figure 4, we group two operations–initial of distance MLS and transition of closer–together. This defines a "intensify" pattern making the camera move closer and closer to the on-screen target. Incidentally, this is exactly what the existing technique, intensify, does in a real film. The syntax of the pattern intesify would be defined as:

```
pattern "intensify" (length>=3){
   [initial: "distance"
            ("target": target1,
            "distance":mls)
   ]
   [next: "closer"
      ("target":target1)
   ]
   [transition:
      ("type":cut)
   ]
}
```

Patterns can be embedded to create complex film structures over long sequences of shots. For, example, given another pattern "shot reverse shot" which is defined as below:

```
pattern "s-reverse s" (length>=2){
   [initial:
      ("type":pov,
      "target":target1)
   ]
   [next:
      ("type":pov,
      "target":target2)
   ]
   [transition:
      ("type":cut)
   ]
}
```

The two patterns can be combined to create a pattern which we name "intensify on pov" which moves closer and closer to two targets, finding shot that fulfill the constraints of initial, next, and transition for both patterns.

```
pattern "intensify on pov" (length>=3){
   [all:
      ("pattern":s-reverse s)
   ]
   [all:
      ("pattern":intensify)
   ]
}
```

From the above explanation, we can see how Patterns allows us to define step-by-step stylistic editing techniques through vocabulary that corresponds to on-screen visible features. Other classical film patterns can also be encoded with Patterns.

4. Applications

Patterns was designed to be both annotative and generative in its application to data-driven cinematography. The language can be used to descibe observations of on-screen properties between frames as patterns, and the extracted patterns can be used in a generative manner paired with a database of framings. The two application scenarios are described with examples in this section.

4.1. From an annotative perspective

Patterns is annotative in that it provides simple vocabulary for describing framing properties in shots and over a number of shots. As shown from above, framing properties can easily be extracted using annotative interfaces, forming databases of framings that can be looked up according to certain properties. Framing properties defined in each operation can then be targeted towards those existing in the database.

As shown in the example of intensify in Figure 4, one can (i) identify an existing pattern in a film (ii) extract the framing properties from key frames, such as actor size, distance, position, etc., and (iii) use operations to identify the changes of the on-screen properties, in this case, decrease of distance.

4.2. From a generative perspective

We implemented a prototype of the generative aspect of the language. We use our language to define well-known techniques and letting the system generate a shot sequence that implements user specifications. The input of the system involves 76 well-annotated framings from four different movies, which serve as a database for operations to search for suitable framings. The second input involved the 3D remaking of a dialogue sequence from *The Shining*, which was imported into a Unity storytelling environment with specified cut points, but completely free cameras.

A number of patterns were specified by the user, including intensify, crossline (an index vector exists as a line that passes through two on screen targets, and crossline is the act of the camera crossing from one side of the index vector to the other, resulting in a switch of the relative positions of two targets on-screen), shot-reverse-shot (an actor filmed looking at something off screen, then cutting to what the actor is looking at), and point of view shot (a shot showing an actor's perspective).

The framing database is linked to a camera placement algorithm designed by [LC12] in order to calculate the suitable camera configurations to realise the framing in the 3D environment. Figure 5 shows how patterns can be applied to the sequence to create complex editing styles.

The database includes multiple possible framings for every operation, and thus allows for a variety of outputs for the same patterns. The current implementation is a basic search

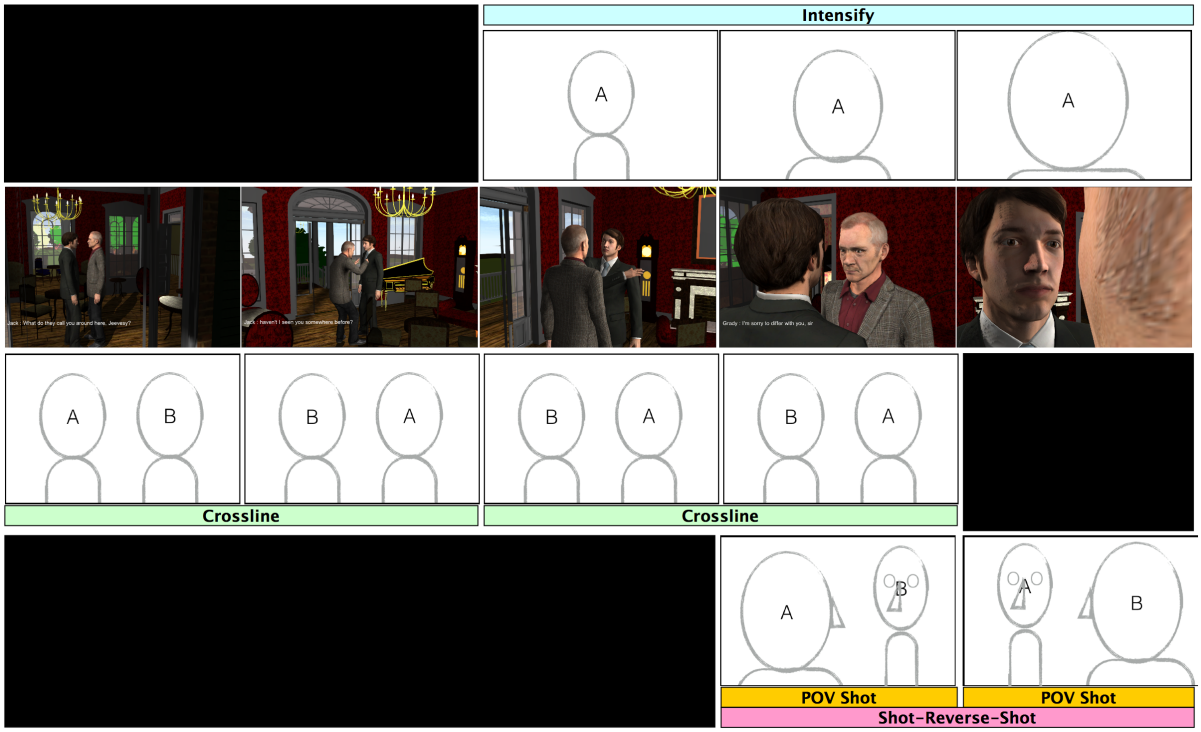

Figure 5: Four types of patterns are combined to generate a sequence for a specified number of cuts.

in the database to match the framing properties that each operation requests. The simple implementation with three overlapping styles displays the potential of the approach.

5. Future Work and Limitations

The pattern language provides a vocabulary for defining the semantic of a sequence of shots using a basic set of vocabulary to describe the structure of operations within the pattern, the framing constraints defined by each operation, and the overall properties of the pattern. We have demonstrated a simple example of how the annotated patterns can be used in a generative system to extract a sequence of framings to build shots that are well-structured.

We envision, with pattern recognition and matching algorithms, there can be strong applications to analytic systems for multimedia content to identify existing camera style. Coupled with machine learning methods, the parameters for the framing restrictions in each operation can also be automatically extracted and weighted, allowing a much smarter selection process for frame selection.

The language also calls for smarter editing tools for 3D environments tailored for prototyping, education, or creativity assisting purposes. Patterns can serve as high level decisions for prototyping the scene quickly, through well-known camera techniques and editing techniques.

The limitations of the language revolves on the difficulty of implementing a robust interpreter for the language. Our current implementation enforces an absolute satisfaction on all constraints. Thus it is not able to evaluate the quality of one framing compared to another, or relax certain constraints to find a best match when no framing satisfies all constraints. Moreover, our current implementation of the language does not observe conflicts between constraints in user-defined patterns, but will simply report no matches in the database.

6. Conclusion and Future Work

In this paper we have proposed Patterns as a language for encoding styles and techniques in cinematographic sequences as patterns. We use this language to annotate framings in existing movies and propose a generative system that relies on our language to define patterns of shots that can be further combined to create complex cinematographic sequences. Examples using Patterns, combined with data-driven approaches, show how the language can adapt to generative systems. The database of patterns can be flexibly extended and adapted to dynamic 3D environments with changing targets and context over complex story scenarios.

References

[ADV02] ADAMS B., DORAI C., VENKATESH S.: Toward automatic extraction of expressive elements from motion pictures: tempo. *IEEE Transactions on Multimedia 4*, 4 (Dec. 2002), 472–481. 3

[AK05] AMERSON D., KIME S.: Real-time cinematic camera control for interactive narratives. In *ACM SIGCHI International Conference on Advances in computer entertainment technology* (2005), ACM Press, pp. 369–369. 1, 3

[BGL98] BARES W. H., GRÉGOIRE J. P., LESTER J. C.: Realtime constraint-based cinematography for complex interactive 3D worlds. In *The National Conference On Artificial Intelligence* (1998), Citeseer, pp. 1101–1106. 1, 3

[BTM00] BARES W. H., THAINIMIT S., MCDERMOTT S.: A Model for Constraint-Based Camera Planning. In *AAAI Spring Symposium* (Stanford, 2000). 1, 3

[CBL11] CANINI L., BENINI S., LEONARDI R.: Affective analysis on patterns of shot types in movies. In *7th International Symposium on Image and Signal Processing and Analysis (ISPA)* (2011), pp. 253–258. 3

[Coh13] COHN N.: Visual narrative structure. *Cognitive science 34* (Apr. 2013), 413–52. 2

[DZ94] DRUCKER S. M., ZELTZER D.: Intelligent camera control in a virtual environment. In *Graphics Interface '94* (1994), pp. 190–199. 3

[ER07] ELSON D. K., RIEDL M. O.: A Lightweight Intelligent Virtual Cinematography System for Machinima Production. In *3rd Conference on Artificial Intelligence and Interactive Digital Entertainment* (2007). 1

[GRC*14] GALVANE Q., RONFARD R., CHRISTIE M., SZILAS N., CAMERA N. S. N.-D.: Narrative-Driven Camera Control for Cinematic Replay of Computer Games. In *Motion in Games 2014* (2014). 3

[GRLC15] GALVANE Q., RONFARD R., LINO C., CHRISTIE M.: Continuity Editing for 3D Animation. In *AAAI Conference on Artificial Intelligence* (Jan. 2015), AAAI Press. 1

[LC12] LINO C., CHRISTIE M.: Efficient Composition for Virtual Camera Control. In *Eurographics / ACM SIGGRAPH Symposium on Computer Animation* (2012). 2, 4, 5

[LCL*10] LINO C., CHRISTIE M., LAMARCHE F., SCHOFIELD G., OLIVIER P.: A Real-time Cinematography System for Interactive 3D Environments. In *Eurographics / ACM SIGGRAPH Symposium on Computer Animation* (2010), no. 1, pp. 139–148. 1

[Mur01] MURCH W.: *In the Blink of an Eye: A Perspective on Film Editing*. new world (for sure) Part 5. Silman-James Press, 2001. 2

[RBLA13] RONFARD R., BOIRON L., LJK I., ART P.: The Prose Storyboard Language. In *AAAI Workshop on Intelligent Cinematography and Editing* (2013). 3

[Ron12] RONFARD R.: A Review of Film Editing Techniques for Digital Games. 2

[TSW*14] TARVAINEN J., SJOBERG M., WESTMAN S., LAAKSONEN J., OITTINEN P.: Content-Based Prediction of Movie Style, Aesthetics, and Affect: Data Set and Baseline Experiments. *IEEE Transactions on Multimedia 16*, 8 (Dec. 2014), 2085–2098. 3

[XWH*11] XU M., WANG J., HASAN M. A., HE X., XU C., LU H., JIN J. S.: Using context saliency for movie shot classification. In *18th IEEE International Conference on Image Processing (ICIP)* (2011), pp. 3653–3656. 3

[Zet07] ZETTL H.: *Sight, sound, motion: Applied media aesthetics*. Wadsworth Publishing Company, 2007. 3